Scalable Gaussian Process Regression Networks

Shibo Li[1] , **Wei Xing**[2] , **Robert M. Kirby**[1,2] and **Shandian Zhe**[1]
[1]School of Computing, University of Utah
[2]Scientific Computing and Imaging Institute, University of Utah
{shibo,kirby,zhe}@cs.utah.edu, wxing@sci.utah.edu

Abstract

Gaussian process regression networks (GPRN) are powerful Bayesian models for multi-output regression, but their inference is intractable. To address this issue, existing methods use a fully factorized structure (or a mixture of such structures) over all the outputs and latent functions for posterior approximation, which, however, can miss the strong posterior dependencies among the latent variables and hurt the inference quality. In addition, the updates of the variational parameters are inefficient and can be prohibitively expensive for a large number of outputs. To overcome these limitations, we propose a scalable variational inference algorithm for GPRN, which not only captures the abundant posterior dependencies but also is much more efficient for massive outputs. We tensorize the output space and introduce tensor/matrix-normal variational posteriors to capture the posterior correlations and to reduce the parameters. We jointly optimize all the parameters and exploit the inherent Kronecker product structure in the variational model evidence lower bound to accelerate the computation. We demonstrate the advantages of our method in several real-world applications.

1 Introduction

Multi-output regression is an important machine learning problem where the critical challenge is to grasp the complex output correlations to enable accurate predictions. Gaussian process regression networks (GPRN) [Wilson *et al.*, 2012] are promising Bayesian models for multi-output regression, which exploit both the structure properties of neural networks and the flexibility of nonparametric function learning by Gaussian processes (GP) [Williams and Rasmussen, 2006]. In GRPNs, the outputs are an adaptive linear projection of a set of latent functions; both the latent functions and projection weights are sampled from independent GPs. In this way, GPRNs can capture input-dependent, highly nonlinear correlations between the outputs, provide heavy-tail predictive distributions and resist overfitting.

However, a critical bottleneck of GPRN is the inference intractability. To address this issue, existing methods use a fully factorized posterior approximation over the latent functions and projection weights to conduct mean-field variational inference [Wilson *et al.*, 2012]. In light of the alternating updates, the posterior covariance matrix for each function and weight can be further parameterized by N (rather than N^2) parameters (N is the number of training samples) [Nguyen and Bonilla, 2013]. In addition, Nguyen and Bonilla (2013) developed nonparametric variational inference that uses a mixture of diagonal Gaussian distributions as the variational posterior of all the latent variables.

Despite the success of the aforementioned approaches, they can suffer from applications with very high-dimensional outputs, which are common in real world, e.g., MRI imaging prediction and physical simulations. First, the fully factorized posterior (and the mixture of diagonal Gaussian) can miss the strong posterior dependencies within the weights and latent function values, resulting in suboptimal quality. Second, the alternating mean-field updates and the optimization of non-parametric inference have $\mathcal{O}(NK^2D)$ and $\mathcal{O}(QN^2KD)$ time complexity respectively, where D, K and Q are the number of the outputs, latent function and mixture components. When D is very large, say, millions, the computational cost can be prohibitively expensive even for moderated N and/or K, say, hundreds.

To overcome these problems, we propose a scalable variational inference algorithm that not only better captures the posterior dependency but also is much more efficient for massive outputs. Specifically, we tensorize the output space so that the projection matrix can be converted into a tensor, based on which we propose a tensor-normal distribution as a joint variational posterior for the projection weights. The tensor-normal posterior not only captures the strong posterior dependencies of the weights, but also requires much less covariance parameters — orders of magnitude less than the output dimension. Similarly, we incorporate a matrix-normal variational posterior for all the latent function values to capture their posterior dependency and save the parameters. Finally, we jointly optimize the variational evidence lower bound (EBLO), where we use the Kronecker product properties to decompose the expensive log determinant and matrix inverse to further accelerate the computation. As the result, our algorithm can linearly scale to all N, D and K, i.e., $\mathcal{O}(NDK)$.

For evaluation, we first examined our method on three

small datasets where the existing GPRN inference approaches are available. Our method shows not only better predictive performance but also a great speed-up. Then we tested our method in two real-world applications with thousands of outputs and the existing GPRN inference algorithms are not feasible. Compared with several state-of-the-art scalable multi-output regression methods, our method almost always achieves significantly better prediction accuracy. Finally, we applied GPRN in a large-scale physical simulation application for one million output prediction. Our method often improves upon the competing approaches by a large margin.

2 Gaussian Process Regression Networks

Let us first introduce the notations and background. Suppose we have a set of N multi-output training examples $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \dots (\mathbf{x}_N, \mathbf{y}_N)\}$, where each output $\mathbf{y}_n$ ($1 \leq n \leq N$) is D dimensional and D can be much larger than N. To model multiple outputs, Gaussian process regression networks (GPRNs) [Wilson *et al.*, 2012] first introduce a small set of K latent functions, $\{f_1(\cdot), \dots, f_K(\cdot)\}$. Each latent function $f_k(\cdot)$ is sampled from a GP prior [Williams and Rasmussen, 2006], a nonparametric function prior that can flexibly estimate various complex functions from data by incorporating (nonlinear) covariance or kernel functions. Next, GPRN introduces a $D \times K$ projection matrix $\mathbf{W}$, where each element $w_{ij} (1 \leq i \leq D, 1 \leq j \leq K)$ is also considered as a function of the input, and sampled from an independent GP prior. Given an input $\mathbf{x}$, the outputs are modelled by

$$\mathbf{y}(\mathbf{x}) = \mathbf{W}(\mathbf{x})[\mathbf{f}(\mathbf{x}) + \sigma_f \boldsymbol{\epsilon}] + \sigma_y \mathbf{z} \tag{1}$$

where $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), \dots, f_K(\mathbf{x})]^\top$, and $\boldsymbol{\epsilon}$ and $\mathbf{z}$ are random noises sampled from the standard normal distribution.

If we view each latent function $f_k(\mathbf{x})$ as an input neuron, GPRN generates the outputs in the same way as neural networks (NNs). Hence GPRN enjoys the structure properties of NNs. Furthermore, GPRN accommodates input-dependent (i.e., non-stationary) correlations of the outputs. Given $\mathbf{W}(\cdot)$, the covariance of arbitrary two outputs $y_i(\mathbf{x}_a)$ and $y_j(\mathbf{x}_b)$ is

$$k_{y_i,y_j}(\mathbf{x}_a, \mathbf{x}_b) = \sum_{k=1}^{K} w_{ik}(\mathbf{x}_a) \kappa_{\hat{f}_k}(\mathbf{x}_a, \mathbf{x}_b) w_{jk}(\mathbf{x}_b) + \delta_{ab}\sigma_y^2$$

where δ_{ab} is 1 if $a = b$ and 0 otherwise, $\kappa_{\hat{f}_k}(\mathbf{x}_a, \mathbf{x}_b) = \kappa_{f_k}(\mathbf{x}_a, \mathbf{x}_b) + \delta_{ab}\sigma_f^2$, and $\kappa_{f_k}(\cdot, \cdot)$ is the covariance (kernel) function for the latent function $f_k(\cdot)$. The output covariance are determined by the inputs via the projection weights $w_{ik}(\mathbf{x}_a)$ and $w_{jk}(\mathbf{x}_b)$. Therefore, the model is able to adaptively capture the complex output correlations varying in the input space. This is more flexible than many popular multi-output regression models [Alvarez *et al.*, 2012] that only impose stationary correlations invariant to input locations.

Now we look into the joint probability of GPRN on the aforementioned training dataset $\mathcal{D}$. Following the original paper [Wilson *et al.*, 2012], we assume all the latent functions share the same kernel $\kappa_f(\cdot, \cdot)$ and parameters $\boldsymbol{\theta}_f$, and all the projection weights the same kernel $\kappa_w(\cdot, \cdot)$ and parameters $\boldsymbol{\theta}_w$. For a succinct representation, we consider a noisy version of each latent function, $\hat{f}_k(\mathbf{x}) = f_k(\mathbf{x}) + \epsilon_k \sigma_f$ where $\epsilon_k \sim \mathcal{N}(0, 1)$. Since f_k is assigned a GP prior, $\hat{f}_k$ also has a GP prior and the kernel is $\kappa_{\hat{f}}(\mathbf{x}_a, \mathbf{x}_b) = \kappa_f(\mathbf{x}_a, \mathbf{x}_b) + \delta_{ab} \cdot \sigma_f^2$. We denote the values of $\hat{f}_k$ at the training inputs $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]^\top$ by $\hat{\mathbf{f}}_k = [\hat{f}_k(\mathbf{x}_1), \dots, \hat{f}_k(\mathbf{x}_N)]^\top$ and projection weight w_{ij} by $\mathbf{w}_{ij} = [w_{ij}(\mathbf{x}_1), \dots, w_{ij}(\mathbf{x}_N)]^\top$. Since $\hat{f}_k(\cdot)$ is sampled from the GP prior, its finite projection follow a multivariate Gaussian prior distribution, $p(\hat{\mathbf{f}}_k | \boldsymbol{\theta}_f, \sigma_f^2) = \mathcal{N}(\hat{\mathbf{f}}_j | \mathbf{0}, \mathbf{K}_{\hat{f}})$ where $\mathbf{K}_{\hat{f}}$ is a kernel matrix and each element $[\mathbf{K}_{\hat{f}}]_{ij} = \kappa_{\hat{f}}(\mathbf{x}_i, \mathbf{x}_j)$. Similarly, the prior of each $\mathbf{w}_{ij}$ is $p(\mathbf{w}_{ij} | \boldsymbol{\theta}_w) = \mathcal{N}(\mathbf{w}_{ij} | \mathbf{0}, \mathbf{K}_w)$ where each $[\mathbf{K}_w]_{ij} = \kappa_w(\mathbf{x}_i, \mathbf{x}_j)$. According to (1), given $\{\mathbf{w}_{ij}\}_{1 \leq i \leq D, 1 \leq j \leq K}$ and $\{\hat{\mathbf{f}}_k\}_{1 \leq k \leq K}$, the observed outputs $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_N]^\top$ are sampled from $p(\mathbf{Y} | \{\mathbf{w}_{ij}\}, \{\hat{\mathbf{f}}_k\}) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{y}_n | \mathbf{W}_n \mathbf{h}_n, \sigma_y^2 \mathbf{I})$ where $\mathbf{W}_n$ is $D \times K$, each $[\mathbf{W}_n]_{ij} = w_{ij}(\mathbf{x}_n)$, and $\mathbf{h}_n = [\hat{f}_1(\mathbf{x}_n), \dots, \hat{f}_K(\mathbf{x}_n)]^\top$. The joint probability then is

$$p(\mathbf{Y}, \{\mathbf{w}_{ij}\}, \{\hat{\mathbf{f}}_k\} | \mathbf{X}, \boldsymbol{\theta}_f, \boldsymbol{\theta}_w, \sigma_f^2, \sigma_y^2) = \prod_{k=1}^{K} \mathcal{N}(\hat{\mathbf{f}}_k | \mathbf{0}, \mathbf{K}_{\hat{f}}) \cdot \prod_{i=1}^{D} \prod_{j=1}^{K} \mathcal{N}(\mathbf{w}_{ij} | \mathbf{0}, \mathbf{K}_w) \prod_{n=1}^{N} \mathcal{N}(\mathbf{y}_n | \mathbf{W}_n \mathbf{h}_n, \sigma_y^2 \mathbf{I}). \tag{2}$$

The inference of GPRN, namely, calculating the exact posterior distribution of the latent function values and projection weights, $\{\mathbf{w}_{ij}\}$ and $\{\hat{\mathbf{f}}_k\}$, and other parameters, is infeasible due to the intractable normalization constant. While we can use Markov-chain Monte-Carlo sampling, it is known to be slow and hard to diagnose the convergence. For more efficient and tractable inference, current approaches [Wilson *et al.*, 2012; Nguyen and Bonilla, 2013] introduce approximate posteriors that are fully factorized over the projection matrix and latent functions and then conduct mean-field variational inference [Wainwright *et al.*, 2008]. Typically, the approximate posterior takes the following form,

$$q(\{\mathbf{w}_{ij}\}, \{\hat{\mathbf{f}}_k\}) = \prod_{k=1}^{K} q(\hat{\mathbf{f}}_k) \prod_{i=1}^{D} \prod_{j=1}^{K} q(\mathbf{w}_{ij}) \tag{3}$$

where each marginal posterior is an N dimensional Gaussian distribution. The mean-field inference enjoys analytical, alternating updates between each $q(\hat{\mathbf{f}}_k)$ and $q(\mathbf{w}_{ij})$. In light of the structure of the updates, the covariance of each posterior can be parameterized by just N rather than N^2 parameters [Nguyen and Bonilla, 2013]. The hyper-parameters $\{\boldsymbol{\theta}_f, \boldsymbol{\theta}_w, \sigma_f^2, \sigma_y^2\}$ are then estimated by gradient-based optimization of the variational evidence lower bound (ELBO). In [Nguyen and Bonilla, 2013], a nonparametric variational inference approach is also developed to better capture multimodality, where the variational posterior is a mixture of diagonal Gaussian distribution,

$$q(\mathbf{u}) = \frac{1}{Q} \sum_{j=1}^{Q} \mathcal{N}(\mathbf{u} | \boldsymbol{\mu}_j, v_j \mathbf{I}) \tag{4}$$

where $\mathbf{u}$ is a vector that concatenate all $\{\mathbf{w}_{ij}\}$ and $\{\hat{\mathbf{f}}_k\}$. The variational parameters $\{\boldsymbol{\mu}_j, v_j\}$ and the hyper-parameters are jointly optimized by maximizing the variational ELBO.

3 Scalable Variational Inference

Despite the success of the existing GPRN inference methods, their performance can be limited by oversimplified posterior structures and they can be computationally too costly for high-dimensional outputs. First, the fully factorized posterior (3) essentially assumes the projection weights and latent functions are mutually independent as in their prior, and completely ignores their strong posterior dependency arising from the coupled data likelihoods. Although the Gaussian mixture approximation (4) alleviates this issue, the diagonal covariance of each component can still miss the abundant posterior correlations. Second, the alternating mean-field updates for (3) and the optimization for (4) in the nonparametric inference take the time complexity $\mathcal{O}(NK^2D)$ and $\mathcal{O}(QN^2KD)$ respectively in each iteration. When D is very large, say, millions, the computation can be prohibitively expensive even for moderated N or K, e.g, hundreds.

To improve both the inference quality and computational efficiency to massive outputs, we develop a scalable variational inference algorithm for GPRN, presented as follows.

3.1 Matrix and Tensor Normal Posteriors

First, to capture the posterior dependency between the latent functions, we use a matrix normal distribution as the joint variational posterior for the $N \times K$ function values $\mathbf{F} = [\hat{\mathbf{f}}_1, \ldots, \hat{\mathbf{f}}_K]^\top$,

$$\begin{aligned} q(\mathbf{F}) &= \mathcal{MN}(\mathbf{F}, \mathbf{M}, \boldsymbol{\Sigma}, \boldsymbol{\Omega}) \\ &= \mathcal{N}(\mathrm{vec}(\mathbf{F})|\mathrm{vec}(\mathbf{M}), \boldsymbol{\Sigma} \otimes \boldsymbol{\Omega}) \end{aligned} \tag{5}$$

where $\otimes$ is the Kronecker product, $\boldsymbol{\Sigma}$ and $\boldsymbol{\Omega}$ are $N \times N$ row covariance and $K \times K$ column covariance matrices, respectively. To ensure the positive definiteness, we further parameterize $\boldsymbol{\Sigma}$ and $\boldsymbol{\Omega}$ by their Cholesky decomposition, $\boldsymbol{\Sigma} = \mathbf{U}\mathbf{U}^\top$ and $\boldsymbol{\Omega} = \mathbf{V}\mathbf{V}^\top$. The matrix Gaussian posterior not only captures the dependency of the function values, but also reduces the number of parameters and computational cost. If we use a factorized posterior over each $\hat{\mathbf{f}}_k$, the number of parameters is $NK + KN(N+1)/2$, including the mean and Cholesky decomposition of the covariance for each k, while our approximate posterior only needs $NK + N(N+1)/2 + K(K+1)/2$ parameters.

Next, we consider the variational posterior of the projection weights $\{\mathbf{w}_{ij}\}$. To obtain a joint posterior yet with compact parameterization, we tensorize the D dimensional output space into an M-mode tensor space, $d_1 \times \ldots \times d_M$ where $D = \prod_{m=1}^{M} d_m$. For simplicity, we set $d_1 = \ldots = d_M = d = \sqrt[M]{D}$. Then we can organize all the weights into an $N \times K \times d_1 \times \ldots \times d_M$ tensor $\mathcal{W}$. To capture the posterior dependencies between all the weights, we introduce a tensor normal distribution — a straightforward extension of the matrix normal distribution — as the variational posterior for $\mathcal{W}$,

$$\begin{aligned} q(\mathcal{W}) &= \mathcal{TN}(\mathcal{W}|\mathcal{U}, \boldsymbol{\Gamma}_1, \ldots, \boldsymbol{\Gamma}_{M+2}) \\ &= \mathcal{N}(\mathrm{vec}(\mathcal{W})|\mathrm{vec}(\mathcal{U}), \boldsymbol{\Gamma}_1 \otimes \ldots \otimes \boldsymbol{\Gamma}_{M+2}) \end{aligned} \tag{6}$$

where $\mathcal{U}$ is the mean tensor, $\{\boldsymbol{\Gamma}_1, \ldots, \boldsymbol{\Gamma}_{M+2}\}$ are covariance matrices in each mode, $\boldsymbol{\Gamma}_1$ is $N \times N$, $\boldsymbol{\Gamma}_2$ is $K \times K$, and $\boldsymbol{\Gamma}_{3:M+2}$ are $d \times d$. To ensure positive definiteness, we parameterize each covariance matrix by its Cholesky decomposition, $\boldsymbol{\Gamma}_m = \mathbf{L}_m\mathbf{L}_m^\top$. Note that to represent the entire $NKD \times NKD$ covariance matrix of $q(\mathcal{W})$, we only need $N(N+1)/2 + K(K+1)/2 + Md(d+1)/2$ parameters which can be even far less than NDK. Take $D = 10^6, N = 100, K = 10$ as an example, the number of the covariance parameters is $0.2\%NKD$ (for $M = 3$). For the mean-field posterior in (3), the number of covariance parameters for all the projection weights is NKD even with the compact representation. Therefore, our tensor normal posterior not only preserves the strong correlations among all the weights, but also saves much more parameters and so computation cost.

Finally, we choose our variational posterior as $q(\mathbf{F}, \mathcal{W}) = q(\mathbf{F})q(\mathcal{W})$. While it stills factorizes over $\mathbf{F}$ and $\mathcal{W}$, the strong correlations of many variables within $\mathbf{F}$ and $\mathcal{W}$ are captured, and hence still improves upon the fully factorized posterior.

3.2 Simplified Variational Evidence Lower Bound

Now, we derive the evidence lower bound (ELBO) with our proposed variational posterior,

$$\mathcal{L} = \mathbb{E}_q[\log \frac{p(\mathbf{Y}, \mathbf{F}, \mathcal{W}|\boldsymbol{\theta}_f, \boldsymbol{\theta}_w, \sigma_f^2, \sigma_y^2)}{q(\mathbf{F}, \mathcal{W})}]$$

We will maximize the ELBO to jointly optimize the variational parameters and hyper-parameters. To accelerate the computation, we further use the properties of the Kronecker product [Stegle *et al.*, 2011] in our variational posteriors ((5) and (6)) to dispose of their full covariances and derive a much simplified bound,

$$\begin{aligned} \mathcal{L} = &-\mathrm{KL}\big(q(\mathcal{W})\|p(\mathcal{W})\big) - \mathrm{KL}\big(q(\mathbf{F})\|p(\mathbf{F})\big) \\ &+ \mathbb{E}_q[\log p(\mathbf{Y}|\mathbf{X}, \mathcal{W}, \mathbf{F})] \end{aligned} \tag{7}$$

where

$$\begin{aligned} &\mathrm{KL}\big(q(\mathcal{W})\|p(\mathcal{W})\big) = \frac{1}{2}\big[\mathrm{tr}(\mathbf{K}_w^{-1}\boldsymbol{\Gamma}_1) \prod_{m=2}^{M+2} \mathrm{tr}(\boldsymbol{\Gamma}_m) + \\ &DK \log|\mathbf{K}_w| + \mathrm{tr}(\mathbf{K}_{\hat{f}}^{-1}\mathbf{U}_1\mathbf{U}_1^\top) - \sum_{m=1}^{M+2} \frac{NDK}{t_m} \log|\boldsymbol{\Gamma}_m|\big], \\ &\mathrm{KL}\big(q(\mathbf{F})\|p(\mathbf{F})\big) = \frac{1}{2}\big[\mathrm{tr}(\mathbf{K}_{\hat{f}}^{-1}\boldsymbol{\Sigma})\mathrm{tr}(\boldsymbol{\Omega}) + \mathrm{tr}(\mathbf{K}_{\hat{f}}^{-1}\mathbf{M}\mathbf{M}^\top) \\ &+ K \log|\mathbf{K}_{\hat{f}}| - (K \log|\boldsymbol{\Sigma}| + N \log|\boldsymbol{\Omega}|)\big], \end{aligned}$$

and

$$\begin{aligned} &\mathbb{E}_q[\log p(\mathbf{Y}|\mathbf{X}, \mathcal{W}, \mathbf{F})] = -ND \log \sigma_y - \sum_{n=1}^{N} \frac{1}{2\sigma_y^2}\big[\mathbf{y}_n^\top\mathbf{y}_n \\ &- 2\mathbf{y}_n^\top\mathbb{E}_q[\mathbf{W}_n]\mathbb{E}_q[\mathbf{h}_n]\big] + \mathrm{tr}(\mathbb{E}_q[\mathbf{W}_n^\top\mathbf{W}_n]\mathbb{E}_q[\mathbf{h}_n\mathbf{h}_n^\top])\big]. \end{aligned}$$

Here $\mathbf{U}_1$ is an $N \times DK$ matrix, obtained by unfolding the mean tensor $\mathcal{U}$ at mode 1, t_m is the dimension of mode m of the tensor $\mathcal{W}$, $\mathbb{E}_q[\mathbf{W}_n]$ is obtained by taking the n-th slice of $\mathcal{U}$ at mode 1 and then reorganize it into a $D \times K$ matrix,

$\mathbb{E}_q[\mathbf{h}_n]$ the n-th row vector of $\mathbf{M}$, and the remaining moments are calculated by

$$\mathbb{E}_q[\mathbf{W}_n^\top \mathbf{W}_n] = \mathbf{\Gamma}_2 \prod_{m=3}^{M+2} \mathrm{tr}(\mathbf{\Gamma}_m) + \mathbb{E}_q[\mathbf{W}_n]^\top \mathbb{E}_q[\mathbf{W}_n],$$
$$\mathbb{E}_q[\mathbf{h}_n \mathbf{h}_n^\top] = \mathbf{\Omega} \cdot \mathrm{diag}(\mathbf{\Sigma}) + \mathbb{E}_q[\mathbf{h}_n]\mathbb{E}_q[\mathbf{h}_n]^\top.$$

As we can see, the computation of the ELBO (7) only involves the covariance matrices at each mode of $\mathcal{W}$, $\mathbf{F}$ and the kernel matrices: $\{\mathbf{\Gamma}_{1:M+2}, \mathbf{\Sigma}, \mathbf{\Omega}, \mathbf{K}_{\hat{f}}, \mathbf{K}_w\}$, which are small even for a very large number of outputs. Hence the computation is much simplified. In addition, due to the compact parameterization, optimizing the ELBO is much easier. We then use gradient-based optimization methods to jointly estimate the parameters of the variational posteriors and the hyper-parameters $\{\sigma_f^2, \sigma_y^2, \boldsymbol{\theta}_f, \boldsymbol{\theta}_w\}$.

3.3 Prediction

Given a new input $\mathbf{x}^*$, we aim to use the estimated variational posterior to predict the output $\mathbf{y}^*$. The posterior mean of $\mathbf{y}^*$ is computed by $\mathbb{E}[\mathbf{y}^*|\mathbf{x}^*, \mathcal{D}] = \mathbb{E}[\mathbf{W}(\mathbf{x}^*)]\mathbb{E}[\hat{\mathbf{f}}(\mathbf{x}^*)]$ where $\hat{\mathbf{f}}(\mathbf{x}^*) = [\hat{f}_1(\mathbf{x}^*), \ldots, \hat{f}_K(\mathbf{x}^*)]^\top$. Each $[\mathbb{E}[\mathbf{W}(\mathbf{x}^*)]]_{ij}$ is computed by $\mathbf{k}_w^* \mathbf{K}_w^{-1} \mathbf{v}_{ij}$ where $\mathbf{k}_w^* = [\kappa_w(\mathbf{x}^*, \mathbf{x}_1), \ldots, \kappa_w(\mathbf{x}^*, \mathbf{x}_N)]$ and $\mathbf{v}_{ij}$ is obtained by first reorganizing $\mathcal{U}$ (the posterior mean of $\mathcal{W}$) to an $N \times K \times D$ tensor $\widehat{\mathcal{U}}$ and then taking the fiber $\widehat{\mathcal{U}}(:, i, j)$. Each $\mathbb{E}(\hat{f}_k(\mathbf{x}^*)) = \mathbf{k}_{\hat{f}}^* \mathbf{K}_{\hat{f}}^{-1} \mathbf{M}(:, k)$ where $\mathbf{k}_{\hat{f}}^* = [\kappa_{\hat{f}}(\mathbf{x}^*, \mathbf{x}_1), \ldots, \kappa_{\hat{f}}(\mathbf{x}^*, \mathbf{x}_N)]$.

The predictive distribution, however, does not have a close form, because the likelihood is non-Gaussian w.r.t the projection weights and latent functions. To address this issue, we can use Monte-Carlo approximations. We can generate a set of i.i.d posterior samples of $\mathbf{W}(\mathbf{x}^*)$ and $\hat{\mathbf{f}}(\mathbf{x}^*)$, denoted by $\{\widetilde{\mathbf{W}}_t, \widetilde{\mathbf{f}}_t\}_{t=1}^T$, and then approximate $p(y^*|\mathbf{x}^*, \mathcal{D}) \approx \frac{1}{T}\sum_{t=1}^T \mathcal{N}(\mathbf{y}^*|\widetilde{\mathbf{W}}_t \widetilde{\mathbf{f}}_t, \sigma_y^2 \mathbf{I})$. Note that when the output dimension is large, the posterior sample of $\mathbf{W}(\mathbf{x}^*)$ can be too costly to generate. We can instead approximate the predictive distribution of each single output y_j^* with the same method.

3.4 Algorithm Complexity

The overall time complexity of our inference algorithm is $\mathcal{O}(N^3 + K^3 + Md^2 + NDK)$. When $D \gg \{N, K\}$ and $Md^2 \leq D$, the complexity is $\mathcal{O}(NDK)$. It is trivial to show that when $3 \leq M \leq \sqrt[3]{D}$, we always have $Md^2 \leq D$. The space complexity is $\mathcal{O}(NKD + ND + N^2 + K^2 + Md^2)$ including the storage of training data, kernel matrices, the variational posterior parameters and the other parameters.

4 Related Work

Many multi-output regression approaches have been proposed and most of them are based on GPs; see an excellent review in [Alvarez *et al.*, 2012]. A classical method is the linear model of coregionalization (LMC) [Goulard and Voltz, 1992], which projects a set of latent functions to the multi-output space; each latent function is sampled from an independent GP. PCA-GP is a popular LCM [Higdon *et al.*, 2008] that identifies the projection matrix by Singular Value Decomposition (SVD). The variants of PCA-GP include KPCA-GP [Xing *et al.*, 2016], IsoMap-GP [Xing *et al.*, 2015], etc. GPRN [Wilson *et al.*, 2012] is another instance of LMC. However, since the projection weights are also sampled from GPs, the prior of the outputs is no longer a GP. Other approaches include convolved GPs [Higdon, 2002; Boyle and Frean, 2005; Alvarez *et al.*, 2019] and multi-task GPs [Bonilla *et al.*, 2007; 2008; Rakitsch *et al.*, 2013]. Convolved GPs generate each output by convolving a smoothing kernel and a set of latent functions. Multi-task GPs define a product kernel over the input features and task dependent features (or free-from task correlation matrices). Despite their success, both types of models might be too costly ($\mathcal{O}((ND)^3)$ or $\mathcal{O}(N^3 + D^3)$ time complexity) for high-dimensional outputs. To mitigate this issue, several sparse approximations have been developed [Alvarez and Lawrence, 2009; Álvarez *et al.*, 2010]. Recently, Zhe *et al.* (2019) introduced latent coordinate features in the tensorized output space to model complex correlations and to predict tensorized outputs. Their method essentially constructs a product kernel with which to simplify the inference. By contrast, our work introduces tensor/matrix-normal variational posteriors to improve GPRN inference and does not assume any special kernel structure in GPRN.

5 Experiment

5.1 Predicting Small Numbers of Outputs

We first evaluated the proposed GPRN inference on five real-world datasets with a small number of outputs: (1) *Jura* [1], the heavy-metal concentration measurements of 349 neighbouring locations in Swiss Jura. Following [Wilson *et al.*, 2012], we predicted 3 correlated concentrations, cadmium, nickel and zinc, given the locations of the measurements. (2)*Equity* [2] [Wilson *et al.*, 2012], a financial datasets that include 643 records of 5 equity indices — NASDAQ, FTSE, TSE, NIKKEI and DJC. The inputs are the 5 indices, and the goal is to predict their 25 pair-wise correlations. (3) *PM2.5*[3], 100 spatial measurements (i.e., outputs) of the particulate matter pollution (PM2.5) in Salt Lake City in July 4-7, 2018. The inputs are time points of the measurements. (4) *Cantilever* [Andreassen *et al.*, 2011], material structures with the maximum stiffness on bearing forces from the right side. The input of each example is the force and the outputs are a 3,200 dimensional vector that represents the stress field that determines the optimal material layout in a 80×40 rectangular domain. (5) *GeneExp*[4], expressions of 4,511 genes (outputs) measured by different microarrays, each of which is described by a 10 dimensional input vector.

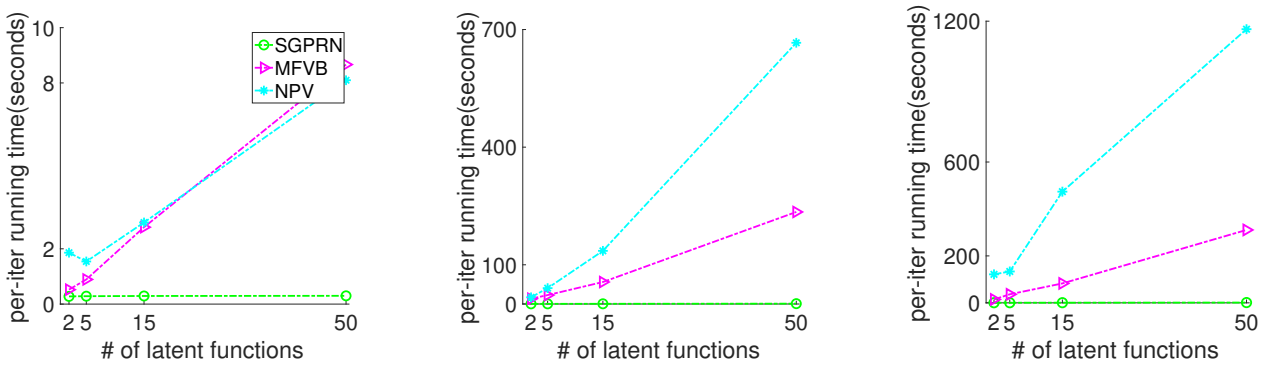
Competing methods. We compared our inference algorithm, denoted by SGPRN, with the following approaches. (1) MFVB — mean-field variational Bayes inference for GPRN [Wilson *et al.*, 2012], and (2) NPV — nonparametric variational Inference for GPRN [Nguyen and Bonilla,

[1] https://rdrr.io/cran/gstat/man/jura.html

[2] https://github.com/davidaknowles/gprn

[3] http://www.aqandu.org/

[4] https://www.synapse.org/#!Synapse:syn2787209/wiki/70350



(a) *Jura* (b) *Equity* (c) *PM 2.5*

Figure 1: Training speed of three GPRN inference algorithms.

2013]. In addition, we compared with three other multi-output GP models that are scalable to high-dimensional outputs: (3) PCA-GP [Higdon *et al.*, 2008] that uses PCA to find the projection matrix in the LMC framework [Goulard and Voltz, 1992] for multi-output regression, (4) KPCA-GP [Xing *et al.*, 2015] and (5) IsoMap-GP [Xing *et al.*, 2016] that use KPCA [Schölkopf *et al.*, 1998] and (5) IsoMap [Balasubramanian and Schwartz, 2002] respectively to identify the projection matrix. We also compared with (5) HOGP [Zhe *et al.*, 2019], high-order GP for regression that introduces latent coordinate features for tensorized output prediction (see Sec. 4 Related Work).

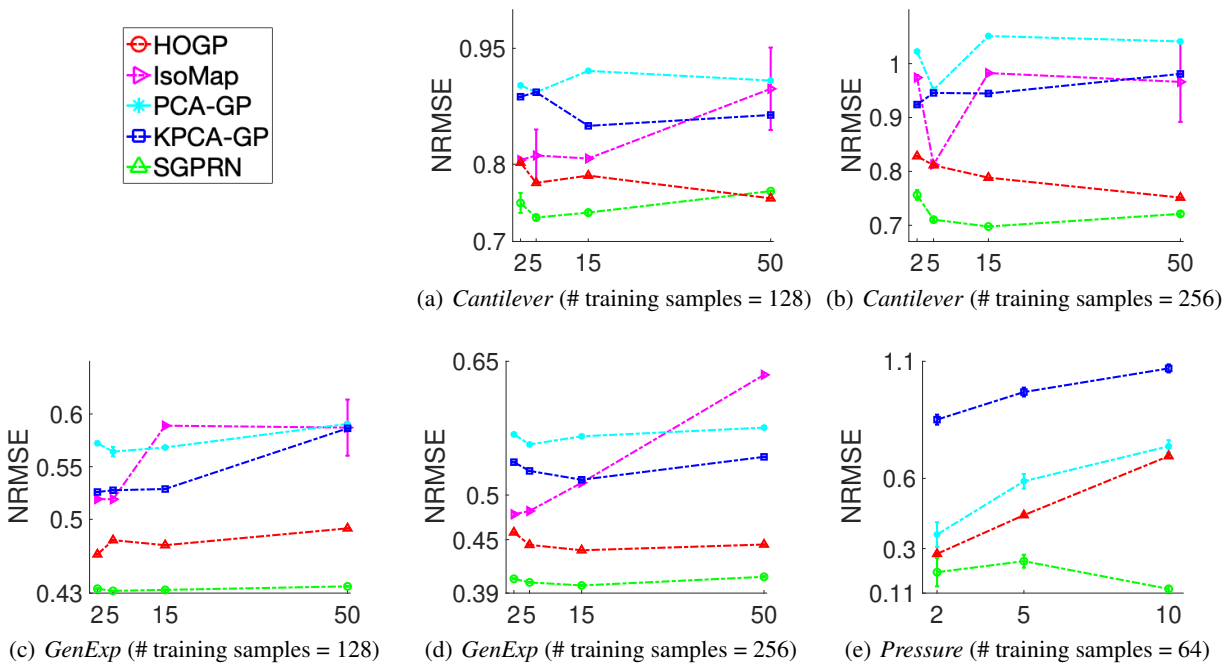
Parameter settings. We implemented our method SGPRN with TensorFlow [Abadi *et al.*, 2016]. We used Adam [Kingma and Ba, 2014] algorithm for gradient-based optimization and the learning rate was set to 10^{-3}. All the competing methods were implemented with MATLAB. For MFVB and NPV, we used the efficient implementation (https://github.com/trungngv/gprn) of the paper [Nguyen and Bonilla, 2013]. We used 2 mixture components for the variational poster in NVP (see (4)). The competing methods used L-BFGS for optimization and the number of iterations was set to 100. We used RBF kernel for all the methods. The input features of all the datasets were normalized, and the kernel parameters (i.e., the length-scale) were initialized to 1. We varied the number of latent functions/features/bases from $\{2, 5, 15, 50\}$.

Comparison with state-of-the-art GPRN inference. We first compared with MFVB and NPV on the three smallest datasets, *Jura*, *Equity* and *PM2.5*, with 2, 25 and 100 outputs respectively. We tested SGRPN, MFVB and NPV on a workstation with 2 Intel(R) Xeon(R) E5-2697 CPUs, 28 cores and 196GB memory. Note that MFVB and NPV are not available on the other datasets (i.e., *Cantilever* and *GeneExp*) — our test shows that they will take extremely long time (weeks or months) to train with 15 and 50 latent functions. We followed the setting of the original paper [Wilson *et al.*, 2012] and [Nguyen and Bonilla, 2013] to only use 2 latent functions. On *Jura*, we randomly split the data into 249 examples for training and 100 for test, on *Equity* 200 for training and 200 for test, and on *PM2.5* 256 for training and 32 for test. In our algorithm, for *Jura* and *Equity*, we used the original output space, and for *PM2.5* we tensorized the output space to be 10×10. We ran our algorithm for $2,000$ epochs to ensure convergence; both MFVB and NPV converged after 100 iterations. We repeated for 5 times and reported the average of the mean absolute error (MAE) and the standard deviation in Table 1. As we can see, our method SGPRN significantly outperforms (p-value < 0.05) both MFVB and NPV on all the three datasets, showing superior inference quality. Note that NPV obtains much bigger MAEs on *PM2.5*. This might be because the optimization of the variational ELBO converged to inferior local maximums.

Jura	
SGPRN	**0.5127**±0.002
MFVB	0.5237±0.001
NPV	0.6088±9e-06
Equity	
SGPRN	**2.5759e-05**±2e-07
MFVB	4.4530e-05±6e-07
NPV	4.7267e-05±9e-18
PM2.5	
SGPRN	**1.07089**±0.02
MFVB	1.3916±0.06
NPV	14.7101±0.14

Table 1: The mean absolute error(MAE) of the three GPRN inference methods. The results were averaged over 5 runs.

Comparison of running time. Next, we compared with MFVB and NPV in terms of the speed. We reported the per-iteration running time of MFVB and NPV in Fig. 1. Since our algorithm ran $2,000$ epochs while MFVB and NPV 100 iterations, we used the average running time of 20 epochs of SGPRN as the per-iteration time for a fair comparison. As we can see, the speed of SGPRN is close to that of MFVB and NPV for very a few latent function (2 and 5). However, with more latent functions, SGPRN becomes much faster. For example, on *PM2.5* with 50 latent functions, SGPRN gains 200x and 785x speed-up as compared with MFVB and NPV.



(a) *Cantilever* (# training samples = 128) (b) *Cantilever* (# training samples = 256)

(c) *GenExp* (# training samples = 128) (d) *GenExp* (# training samples = 256) (e) *Pressure* (# training samples = 64)

Figure 2: The normalized root-mean-squared error(NRMSE) of all the methods on three datasets. The results were averaged over 5 runs. The x-axis represents the number of latent functions of SGPRN, features of HOGP, and bases of IsoMap-GPR, KPCA-GPR, and PCA-GPR.

This is consistent with their time complexities, {MFVB: $\mathcal{O}(NK^2D)$, NPV: $\mathcal{O}(QN^2KD)$, SGPRN:$\mathcal{O}(NKD)$}.

Comparison with other methods. Next, we compared with other scalable multi-output GP regression models. To this end, we used *Cantilever* and *GeneExp* datasets. For our algorithm and HOGP, we tensorized the output space of *Cantilever* to 80×40 and *GeneExp* to 347×13. On each dataset, we randomly chose $\{128, 256\}$ examples for training and 100 from the remaining set for test. We repeated this procedure for 5 times and reported the average normalized root-mean-square error (NRMSE) and the standard deviation of each method in Fig. 2. As we can see, SGPRN significantly outperforms the competing methods (p-value < 0.05) in all the cases except when training on *Cantilever* with 128 examples and 50 latent functions , SGPRN was a little worse than PCA-PG (Fig. 2a). Therefore, it demonstrates the advantage of GPRN in predictive performance, which might be due to its capability of capturing non-stationary output dependencies.

5.2 Large-Scale Physical Simulations for Lid-Driven Cavity Flows

Finally, we applied SGPRN in a large-scale physical simulation application. Specifically, we trained GPRN to predict a one-million dimensional pressure field for lid-driven cavity flows [Bozeman and Dalton, 1973], which include turbulent flow inside the cavity. The simulation of the field is done by solving the Navier-Stoke equations that are known to have complicated behaviours under large Reynolds numbers. We used a fine-grained mesh to ensure the numerical solver to converge. The input of each simulation example is a 5 dimensional vector that represents a specific boundary condition. The computation for each simulation is very expensive, so we only collected 96 examples. Hence, this is a typical "large D, small N" problem. We randomly split the dataset into 64 training and 32 test examples, and then ran SGPRN, PCA-GP, KPCA-GP and IsoMAP-GP. For SGPRN, we tensorized the one million outputs into a $100 \times 100 \times 100$ tensor. We varied the number of latent functions from $\{2, 5, 10\}$. We repeated the training and test procedure for 5 times and showed the average normalized root-mean-square error (NRMSE) of each method in Fig.2(e). As we can see, our method significantly outperforms all the competing approaches, by a large margin especially when using 5 and 10 latent functions. The results further demonstrate the advantage of GPRN for large-scale multi-out regression tasks when it is available.

6 Conclusion

We have proposed a scalable variational inference algorithm for GPRN, a powerful Bayesian multi-output regression model. Our algorithm not only improves the inference quality upon the existing GPRN inference methods but also is much more efficient and scalable to a large number of outputs. In the future work, we will continue to explore GPRN in large-scale multi-output regression applications.

Acknowledgments

This work has been supported by DARPA TRADES Award HR0011-17-2-0016 and NSF IIS-1910983.

References

[Abadi *et al.*, 2016] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[Alvarez and Lawrence, 2009] Mauricio Alvarez and Neil D Lawrence. Sparse convolved gaussian processes for multi-output regression. In *Advances in neural information processing systems*, pages 57–64, 2009.

[Álvarez *et al.*, 2010] Mauricio Álvarez, David Luengo, Michalis Titsias, and Neil Lawrence. Efficient multioutput gaussian processes through variational inducing kernels. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 25–32, 2010.

[Alvarez *et al.*, 2012] Mauricio A Alvarez, Lorenzo Rosasco, Neil D Lawrence, et al. Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning*, 4(3):195–266, 2012.

[Alvarez *et al.*, 2019] Mauricio Alvarez, Wil Ward, and Cristian Guarnizo. Non-linear process convolutions for multi-output gaussian processes. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1969–1977, 2019.

[Andreassen *et al.*, 2011] Erik Andreassen, Anders Clausen, Mattias Schevenels, Boyan S Lazarov, and Ole Sigmund. Efficient topology optimization in matlab using 88 lines of code. *Structural and Multidisciplinary Optimization*, 43(1):1–16, 2011.

[Balasubramanian and Schwartz, 2002] Mukund Balasubramanian and Eric L Schwartz. The isomap algorithm and topological stability. *Science*, 295(5552):7–7, 2002.

[Bonilla *et al.*, 2007] Edwin V Bonilla, Felix V Agakov, and Christopher KI Williams. Kernel multi-task learning using task-specific features. In *Artificial Intelligence and Statistics*, pages 43–50, 2007.

[Bonilla *et al.*, 2008] Edwin V Bonilla, Kian M Chai, and Christopher Williams. Multi-task gaussian process prediction. In *Advances in neural information processing systems*, pages 153–160, 2008.

[Boyle and Frean, 2005] Phillip Boyle and Marcus Frean. Dependent gaussian processes. In *Advances in neural information processing systems*, pages 217–224, 2005.

[Bozeman and Dalton, 1973] James D Bozeman and Charles Dalton. Numerical study of viscous flow in a cavity. *Journal of Computational Physics*, 12(3):348–363, 1973.

[Goulard and Voltz, 1992] Michel Goulard and Marc Voltz. Linear coregionalization model: tools for estimation and choice of cross-variogram matrix. *Mathematical Geology*, 24(3):269–286, 1992.

[Higdon *et al.*, 2008] Dave Higdon, James Gattiker, Brian Williams, and Maria Rightley. Computer model calibration using high-dimensional output. *Journal of the American Statistical Association*, 103(482):570–583, 2008.

[Higdon, 2002] Dave Higdon. Space and space-time modeling using process convolutions. In *Quantitative methods for current environmental issues*, pages 37–56. Springer, 2002.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Nguyen and Bonilla, 2013] Trung Nguyen and Edwin Bonilla. Efficient variational inference for gaussian process regression networks. In *Artificial Intelligence and Statistics*, pages 472–480, 2013.

[Rakitsch *et al.*, 2013] Barbara Rakitsch, Christoph Lippert, Karsten Borgwardt, and Oliver Stegle. It is all in the noise: Efficient multi-task gaussian process inference with structured residuals. In *Advances in neural information processing systems*, pages 1466–1474, 2013.

[Schölkopf *et al.*, 1998] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[Stegle *et al.*, 2011] Oliver Stegle, Christoph Lippert, Joris M Mooij, Neil D Lawrence, and Karsten Borgwardt. Efficient inference in matrix-variate gaussian models with\iid observation noise. In *Advances in neural information processing systems*, pages 630–638, 2011.

[Wainwright *et al.*, 2008] Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

[Williams and Rasmussen, 2006] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006.

[Wilson *et al.*, 2012] Andrew Gordon Wilson, David A Knowles, and Zoubin Ghahramani. Gaussian process regression networks. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 1139–1146. Omnipress, 2012.

[Xing *et al.*, 2015] Wei Xing, Akeel A Shah, and Prasanth B Nair. Reduced dimensional gaussian process emulators of parametrized partial differential equations based on isomap. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2174):20140697, 2015.

[Xing *et al.*, 2016] WW Xing, V Triantafyllidis, AA Shah, PB Nair, and Nicholas Zabaras. Manifold learning for the emulation of spatial fields from computational models. *Journal of Computational Physics*, 326:666–690, 2016.

[Zhe *et al.*, 2019] Shandian Zhe, Wei Xing, and Robert M Kirby. Scalable high-order gaussian process regression. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2611–2620, 2019.